ELSEVIER

Discrete Applied Mathematics 114 (2001) 43–59

DISCRETE APPLIED MATHEMATICS

Average case complexity for finite Boolean functions ☆

A.V. Chashkin

Faculty of Mathematics and Mechanics, Moscow State University, Vorob'evy Gory, 119899 Moscow, Russia

Abstract

The average time of computing Boolean functions by straight-line programs with a conditional stop is considered. A straight-line program consists of operators of two types. Every operator of the first type computes a binary Boolean function whose arguments are either the values computed by preceding operators or the values of the input variables. Every operator of the second type either terminates execution of the program or commands that the next operator is executed. A measure of the complexity of such programs is the average execution time over all tuples of the input variables. The complexity of computing almost all complete Boolean functions and almost all partial Boolean functions by such programs is proved to coincide (up to a multiplicative constant) with conventional circuit complexity. Moreover, these complexities are proved to differ (up to a multiplicative constant) by a factor of n for almost all n-place Boolean functions that are equal to 1 on n^c tuples. It is proved that there exist Boolean functions of n variables whose average execution time is less (up to a multiplicative constant) by a factor of $(2^n/n)^{1/2}$ than the time required to compute them by conventional straight-line programs.

Keywords: Boolean function; Complexity

1. Introduction

In this paper, we study the complexity of computing Boolean functions by straight-line programs with a conditional stop. Each such a program is a sequence of operators of two types. Every operator of the first type computes a binary Boolean function whose arguments are either the values computed by preceding operators or the input variables. Operators of the second type may terminate the execution of the program. The result of a second-type operator is determined by the values computed at specified two preceding steps. For each particular operator, the index numbers of these steps are fixed and may be different for different operators.

☆ Translated from Discrete Anal. Oper. Res., Novosibirsk 4(1) (1997) 60–78. This work was supported by the Russian Foundation for Basic Research, project No. 96-01-01068.

E-mail address: chash@online.ru, chash@mech.math.msu.su (A.V. Chashkin).

0166-218X/01/$ - see front matter
PII: S0166-218X(00)00360-7

If the first argument of a second-type operator equals unity, then the program terminates, and the value of the argument of the second operator is declared to be the value of the Boolean function on the tuple of variables considered. Otherwise, the next operator is executed. If the last operator of the program is a first-type operator and the program was not terminated at the preceding steps, then the result of the program is the value computed by the last operator. Unlike conventional straight-line programs (circuits), which require the same number of steps to compute a function on different arguments, the execution time of a program may be different for different arguments. Thus, the natural measure of the complexity of such programs is the average execution time over all possible arguments.

For this measure, we derive formulas (exact up to a multiplicative constant) for the Shannon function in the classes of all Boolean functions, of Boolean functions with a polynomial number of ones, and of partial Boolean functions. It is found (Theorem 1) that the average time required to compute almost every Boolean function by such programs is less by a constant factor than the time required for conventional straight-line programs, i.e., the possibility of an early stop of computations decreases the average execution time by at most a constant factor. A similar result is established in Theorem 5 for almost all partial Boolean functions. On the other hand, we show (Theorem 2) the existence of functions of n variables for which the average execution time is less by a factor of $(2^n/n)^{1/2}$ (up to a multiplicative constant) than the time required for computation by conventional straight-line programs. Finally, we prove (Theorem 4) that the average execution time of almost every Boolean function of n variables that equals 1 on n^c tuples is less by almost a factor of n than the maximum execution time.

2. Basic definitions

Let $B' = \{f : \{0,1\}^2 \to \{0,1\}\}$ be the set of all unary and binary Boolean functions, $\pi : \{0,1\}^2 \to \{0,1\}^2$ be the binary identity Boolean operator, $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables, and let $B = B' \cup \pi$. A *straight-line program* with a conditional stop is a sequence $P = p_1 p_2 \ldots p_s$ whose members are the operators $p_i = f_i(p_{i,1}, p_{i,2})$, where $f_i \in B$, $p_{i,1} = p_k$, $p_{i,2} = p_l$, $f_k, f_l \in B' \cup X$, and $k, l < i$. An operator p_i is called a *first-type operator*, or a functional operator, if $f_i \in B'$. An operator p_i is called a *second-type operator*, or a stop operator, if $f_i = \pi$. The *complexity* $L(P)$ of program P is the number of operators in P. Put $n(p_i) = i$, i.e., $n(p)$ is the index number of the operator p in P. For functional operators in P their values on an arbitrary tuple x are defined by induction as follows. Let $p_1(x) = f_1(x)$ for the first operator and $p_i(x) = f_i(p_{i,1}(x), p_{i,2}(x))$ for any $i > 1$. Suppose that P consists of exactly k second-type operators. Let q_i denote the ith second-type operator in P, and $q_{i,1}$ and $q_{i,2}$ denote the first and second arguments of this operator, respectively, i.e., the operators $p_{n(q_i),1}$ and $p_{n(q_i),2}$. The result of P on an input tuple x, denoted by $P(x)$,

is defined as

$$P(x) = q_{1,1}(x)q_{1,2}(x) \vee \bar{q}_{1,1}(x)(q_{2,1}(x)q_{2,2}(x) \vee \cdots$$

$$\vee \bar{q}_{k-1,1}(x)(q_{k,1}(x)q_{k,2}(x) \vee \bar{q}_{k,1}(x)p_{L(P)}(x))\ldots).$$

Suppose that P is executed on an input tuple x. The execution time $T_P(x)$ of P on x is the minimum $n(q_j)$ whenever $q_{j,1} = 1$. It is easy to see that $P(x)$ is independent of the operators whose index numbers are greater than $n(q_j)$. Therefore, we can say that the program P terminates after $n(q_j)$ has been computed, and $T_P(x)$ is equal to the number of operators executed before the program terminates. The average execution time P is defined by $T(P) = 2^{-n}\sum T_P(x)$, where the summation is taken over all binary tuples of length n. If a Boolean function f satisfies $P(x) = f(x)$ for an arbitrary tuple of variables x, then the program P is said to implement (or compute) the function f. The *complexity* $L(f)$ of f is the complexity of the shortest program that implements f. The *average execution time* of f is defined as $T(f) = \min T(P)$, where the minimum is taken over all programs implementing f. Let A be a set of Boolean functions. The Shannon function on A is defined as $T_A(n) = \max T(f)$, where the maximum is taken over all functions of n variables in A. When A is the set of all Boolean functions, the index A will be omitted.

It is easy to show that programs consisting of only first-type operators are isomorphic to circuits. Therefore, all results derived for upper bounds on circuit complexity are also valid for the programs in question. Moreover, using this isomorphism, it is easy to deduce some nontrivial issues.

Hereafter, we assume that the number of variables of Boolean functions is sufficiently large. The letters c and c_i, $i = 1, 2, \ldots$, denote suitable constants, and log denotes the logarithm to base 2.

3. Arbitrary Boolean functions

Denote by $N(L_1, L_2, n)$ the number of distinct programs, each implementing a Boolean function of n variables and consisting of at most L_1 first-type operators and at most L_2 second-type operators. Furthermore, let $N(L, n)$ denote the number of distinct circuits each consisting of at most L gates and implementing a Boolean function of n variables over the basis of all binary functions.

Lemma 1. *For any* L_1, L_2, *and* n,

$$N(L_1, L_2, n) \leqslant N(L_1 + 3L_2, n).$$

Proof. To prove the lemma, it is sufficient to model the program by a circuit. Let us show that this can be done so that each first-type operator is associated with only one gate of the circuit, and each second-type operator is associated with three gates of the circuit.

Let $P=(p_1,\ldots,p_m)$ be a program computing a Boolean function $f(x_1,\ldots,x_n)$. We divide P into r subprograms $P_i=(p_{l_{i-1}+1},\ldots,p_{l_i})$, $1\leqslant i\leqslant r$, so that the last subprogram P_r consists of only first-type operators and each of the other subprograms contains exactly one second-type operator $p_{l_i}=q_i$, which is the last operator in the subprogram. Suppose that the first-type operators in P_i are modeled by a circuit S_i. For any i, $1\leqslant i\leqslant r-1$, the circuit S_i has two outputs, which are the outputs of the gates corresponding to the operators $q_{i,1}$ and $q_{i,2}$, and the circuit S_r has one output, which is the output of the gate corresponding to the operator $p_{l_r}=p_{L(P)}$. The construction of S_i is easy because of the above isomorphism between circuits and programs consisting of only first-type operators. Let S_{r+1} be a circuit implementing the function

$$h=x_1y_1\vee\bar{y}_1(x_2y_2\vee\ldots(x_{r-1}y_{r-1}\vee\bar{y}_{r-1}x_r)\ldots).$$

Linking the inputs of S_{r+1} to the outputs of S_i, $1\leqslant i\leqslant r$, we obtain a circuit implementing $f(x_1,\ldots,x_n)$. Obviously, the size of S_{r+1} over the basis of all binary Boolean function is not greater than $3(r-1)$. Thus, we have described how the program P is modeled by a circuit when each first-type operator is associated with one gate of the circuit, and each second-type operator is associated with three gates of the circuit. Lemma 1 is proved. □

Lemma 2 (Lupanov [2]). *For any L and n,*

$$N(L,n)\leqslant(c_1(L+n))^{L+n+2}.$$

Lemmas 1 and 2 imply the following result.

Lemma 3. *For any L_1,L_2,n,*

$$N(L_1,L_2,n)\leqslant(c_1(L_1+3L_2+n))^{L_1+3L_2+n+2}.$$

Let f be an arbitrary Boolean function of n arguments, and P be a program implementing f. Every Boolean tuple x of length n, which is regarded as the binary representation of a positive integer, is associated with its number $N_P(x)$ such that $1\leqslant N_P(x)\leqslant 2^n$; $N_P(x)<N_P(y)$ if $T_P(x)<T_P(y)$; and $N_P(x)<N_P(y)$ if $T_P(x)=T_P(y)$ and $x<y$.

Theorem 1. *There exist constants c_2 and c_3 such that*

$$c_2\frac{2^n}{n}\leqslant T(n)\leqslant c_3\frac{2^n}{n}.$$

Proof. The upper bound for $T(n)$ follows from the upper bound for circuit size [2]. The lower bound will be proved by contradiction. Let f be an arbitrary Boolean function of n variables. By assumption, we then have

$$T(f)\leqslant\frac{c_4 2^n}{n},\tag{1}$$

where c_4 is a sufficiently small constant, to be specified later. Let P be a program computing f and satisfying (1). Choose x_0 such that $N_P(x_0) = 2^{n-1}$. Obviously, we have

$$T_P(x_0) < \frac{c_4 2^{n+1}}{n}, \tag{2}$$

since otherwise

$$T_P(f) \geqslant \frac{1}{2^n} \sum_{x|N_P(x)\geqslant 2^{n-1}} T_P(x) > \frac{c_4 2^n}{n}.$$

We now estimate the number of functions that satisfy (1). To this end, we show how such a function f can be uniquely determined using the program P. Let P' be an initial subprogram of P that is sufficient to compute f on the tuples with numbers not exceeding 2^{n-1}. Obviously, P' determines the values of f on all such tuples. Since the other tuples are uniquely determined by P' and, moreover, have been numbered, the function f will be determined completely if we indicate the binary vector of length 2^{n-1} that consists of the values of f on these tuples. We next find an upper bound on the number M of distinct subprograms P'. Since $T_P(x_0) > n+2$ for large enough n, it follows from Lemma 3 and (2) that

$$M \leqslant (4c_1 T_P(x_0))^{4T_P(x_0)} \leqslant (4c_1 c_4 n^{-1} 2^{n+1})^{4c_4 n^{-1} 2^{n+1}} \leqslant 2^{c_4 c_5 2^n}.$$

Therefore, the total number of functions satisfying (1) does not exceed $2^{c_4 c_5 2^n} 2^{2^{n-1}}$. Taking $c_4 < 1/(2c_5)$, we obtain $2^{c_4 c_5 2^n} 2^{2^{n-1}} < 2^{2^n}$. This means that there exists a function f contradicting (1). Theorem 1 is proved. □

Let us find out how the average execution time for a particular Boolean function can differ from its complexity, i.e., from the time required to compute this function by a conventional straight-line program. Put

$$m(n) = \max(L(f)/T(f)),$$

where the maximum is taken over all Boolean functions of n variables.

Theorem 2. *There exist constants c_6 and c_7 such that*

$$c_6(2^n/n)^{1/2} \leqslant m(n) \leqslant c_7(2^n/n)^{1/2}.$$

Proof. Let $k = \lceil (n + \log n)/2 \rceil$, and let g be the most complicated Boolean function of k variables, i.e., such that $L(g) = \Theta(2^n/n)^{1/2}$. Consider the function $f(x_1, \ldots, x_n) = x_{k+1} \& \cdots \& x_n \& g(x_1, \ldots, x_k)$. It is easy to see that $L(f) = \Theta(2^n/n)^{1/2}$ and $T(f) = \mathrm{O}(1)$. Hence, $m(n) \geqslant c_6(2^n/n)^{1/2}$.

Now let us show that $m(n) \leqslant c_7(2^n/n)^{1/2}$. Let f be an arbitrary Boolean function of n variables, and P be a program that computes f in the minimum average time.

Put $k = \lfloor (n + \log n)/2 \rfloor$. Consider a tuple x such that $N_P(x) = 2^n - 2^k$. It is easy to see that

$$2^{k-n} T_P(x) < T(f). \tag{3}$$

Furthermore, let $\tilde{f}$ be a partial Boolean function defined on all tuples y_i such that $N_P(y_i) > N_P(x)$, and coinciding on them with f. Since $2^k = \Theta(2^n n)^{1/2}$, it follows from [1,3] that there exists a circuit S implementing $\tilde{f}$ and such that

$$|S| = \mathrm{O}(2^n/n)^{1/2}. \tag{4}$$

We now describe a program P' computing f. First, the program P computing f in the minimum average time is used to compute f on the tuples y such that $N_P(y) \leqslant N_P(x)$. Since $2^{n-k} = \mathrm{O}(2^n/n)^{1/2}$, it follows from (3) that the computation of f on these tuples requires at most

$$H_1 = \mathrm{O}(2^n/n)^{1/2} T(f) \tag{5}$$

operators. To compute f on the remaining tuples, we use a circuit implementing the function $\tilde{f}$. It clearly follows from (4) that the program modeling S consists of at most

$$H_2 = \mathrm{O}(2^n/n)^{1/2} \tag{6}$$

operators.

Thus, (5) and (6) imply that the complexity $L(P')$ of the program P' is not greater (up to a multiplicative constant) than

$$(2^n/n)^{1/2} T(f) + (2^n/n)^{1/2} \leqslant 2(2^n/n)^{1/2} T(f).$$

Since $L(f) \leqslant L(P')$, there is constant c_7 such that

$$L(f)/T(f) \leqslant c_7 (2^n/n)^{1/2}.$$

Theorem 2 is proved. □

Theorem 3. *For any Boolean function f, there exists a program P computing f and such that*

$$c_8 T(f) L(f) \leqslant T(P) L(P) \leqslant c_9 T(f) L(f).$$

Proof. Let P be a program that computes f in the minimum average time. Let x be the tuple with minimum number such that $T_P(x) \geqslant 2L(f)$ (if such a tuple does not exist, then the assertion of the theorem is trivial because of $T(f) = T(P) \leqslant L(P) \leqslant 2L(f)$). Clearly, $T_P(x) \leqslant 3L(f)$ because otherwise there would exist $L(f) + 1$ consecutive first-type operators in the program P, which contradicts the assumption that P is minimal. The average execution time of P can be represented as follows:

$$T(P) = 2^{-n} \left(\sum_{N_P(y) < N_P(x)} T_P(y) + \sum_{N_P(y) \geqslant N_P(x)} T_P(y) \right)$$

$$= T_1 + T_2 \geqslant T_1 + 2L(f)(2^n - N_P(x) + 1) 2^{-n}.$$

We transform P by replacing the operators with numbers greater than $T_P(x)$ by a program modeling the minimal circuit. The complexity and the average execution time of the new program P' satisfy the inequalities

$$L(P') \leqslant T_P(x) + L(f) \leqslant 4L(f),$$

$$T(P') \leqslant 2^{-n}\left(\sum_{N_P(y)<N_P(x)} T_{P'}(y) + \sum_{N_P(y)\geqslant N_P(x)} T'_P(y)\right)$$
$$= T'_1 + T'_2 \leqslant T'_1 + 4L(f)(2^n - N_P(x) + 1)2^{-n}$$
$$\leqslant 2T'_1 + 4L(f)(2^n - N_P(x) + 1)2^{-n}.$$

Since $T'_1 = T_1$, we have $T(P') \leqslant 2T(P)$. Theorem 3 is proved. □

4. Functions with a small number of ones

In what follows, we consider Boolean functions of n variables that take values 1 on at most n^c tuples, where $c \geqslant 3$. Let

$$T_c(n) = \max T(f),$$

where the maximum is taken over all Boolean functions of n variables, each function being equal to 1 exactly on n^c tuples (n^c is supposed to be an integer). According to [2], the Shannon function $L_c(n)$ for the circuit complexity of such functions is

$$L_c(n) = \Theta\left(\frac{n^{c+1}}{\log n}\right). \tag{7}$$

In Theorem 4 below, we establish the order of the function $T_c(n)$. A comparison of (7) and Theorem 4 shows that the average and maximum execution times for such functions may differ by a factor of n.

Lemma 4. *Let $D_1 \subset D_2 \subset \{0,1\}^n$ and $m = \lfloor \log |D_2| + 2 \rfloor$. Then there exists a linear operator $F : D_2 \to \{0,1\}^m$ such that*

$$|\{(x,y) \mid x \in D_1,\ y \in D_2 \setminus D_1,\ F(x) = F(y)\}| \leqslant |D_1|/2.$$

Proof. Denote by $F(n,m)$ the set of all linear operators $F : \{0,1\}^n \to \{0,1\}^m$. Obviously, $|F(n,m)| = 2^{nm}$. It is easy to see that for any distinct tuples x, y in $\{0,1\}^n$, there exist 2^{n-1} linear functions f such that $f(x) = f(y)$. Therefore, in $F(n,m)$ there exist 2^{nm-m} distinct operators F such that $F(x) = F(y)$. This implies that

$$2^{-nm} \sum_{x \in D_1, y \in D_2 \setminus D_1} 2^{nm-m} = 2^{-m}|D_1||D_2 \setminus D_1|$$

is the average number of pairs (x,y) on which the values of an operator in $F(n,m)$ are the same. Therefore, there exists an operator F in $F(n,m)$ such that $F(x) = F(y)$ on at most

$$2^{-m}|D_1||D_2 \setminus D_1| < \frac{1}{2|D_2|}|D_1||D_2 \setminus D_1| < \frac{1}{2}|D_1|$$

pairs (x,y), $x \in D_1$, $y \in D_2 \setminus D_1$. Lemma 4 is proved. □

The weight $w(f)$ of a function $f:D \to \{0,1\}$ is the number of tuples in D on which f is equal to 1.

Lemma 5. *Let $c \geqslant 2$, $D \subset \{0,1\}^n$, $|D| \geqslant 10n^c$, and let $f:D \to \{0,1\}$ be a function such that $n^c \leqslant w(f) \leqslant \frac{1}{10}|D|$. Then there exists a function $g:D \to \{0,1\}$ such that*

(a) $g \geqslant f$,

(b) $w(g) \leqslant \frac{3}{2}w(f)$,

(c) $$L(g) \leqslant \frac{\log\binom{4|D|}{w(g)}}{\log\log\binom{4|D|}{w(g)}}(1+o(1)).$$

Proof. Put $D_2 = D$ and $D_1 = \{x \in D_2 \mid f(x) = 1\}$. Let F be the linear operator from Lemma 4. The function $g':\{0,1\}^m \to \{0,1\}$ is defined as $g'(x) = \max f(y)$, where the maximum is taken over all $y \in D_2$ such that $x = F(y)$. Furthermore, the function $g:D \to \{0,1\}$ is defined as $g(x) = \max f(y)$, where the maximum is taken over all $y \in D_2$ such that $F(x) = F(y)$. Obviously, $g(x) = g'(F(x))$ and $g \geqslant f$, i.e., (a) and (b) hold. Therefore, $L(g) \geqslant L(g') + L(F)$. By the well-known result of Lupanov [2] on the circuit size of functions with a small number of ones, we have

$$L(g') \leqslant \frac{\log\binom{4|D|}{w(g)}}{\log\log\binom{4|D|}{w(g)}}(1+o(1)),$$

moreover, $L(F) \leqslant O((n\log|D_2|)/\log n)$. It follows that if $n^c \leqslant w(f) \leqslant \frac{1}{10}|D|$, where $c \geqslant 2$, then (c) holds.

The definition of g implies that

$$w(g) = \sum_{x \in D_2} \max_{F(x)=F(y)} f(y) = \sum_{x \in D_1} f(x) + \sum_{x \in D_2 \setminus D_1} \max_{F(x)=F(y)} f(y)$$

$$\leqslant w(f) + |\{(x,y) \mid x \in D_1,\ y \in D_2 \setminus D_1,\ F(x) = F(y)\}| \leqslant \frac{3}{2}|D_1|.$$

Lemma 5 is proved. □

Lemma 6. *Let $c \geqslant 3$ and $|D| \geqslant 10n^c$. Then any function $f:D \to \{0,1\}$ such that $n^c \leqslant w(f) < |D|/10$ satisfies the inequality*

$$L(f) \leqslant \frac{4\log\binom{4|D|}{\lfloor 3w(f)/2 \rfloor}}{\log\log\binom{4|D|}{\lfloor 3w(f)/2 \rfloor}}(1+o(1)).$$

Proof. Let f_1 be the function from Lemma 5. Then

$$w(f_1) \leqslant \tfrac{3}{2} w(f), \tag{8}$$

$$f_1 \geqslant f, \tag{9}$$

$$L(f_1) \leqslant \frac{\log \binom{4|D|}{w(f_1)}}{\log\log \binom{4|D|}{w(f_1)}}(1 + o(1)). \tag{10}$$

It follows from (8) and (9) that

$$w(f \oplus f_1) \leqslant \tfrac{1}{2} w(f). \tag{11}$$

If $w(f \oplus f_1) > n^{c-1}$, then applying Lemma 5 to $f \oplus f_1$ gives a new function f_2 such that

$$f_2 \geqslant f \oplus f_1, \tag{12}$$

$$w(f_2) \leqslant \tfrac{3}{2} w(f \oplus f_1), \tag{13}$$

$$L(f_2) \leqslant \frac{\log \binom{4|D|}{w(f_2)}}{\log\log \binom{4|D|}{w(f_2)}}(1 + o(1)). \tag{14}$$

It follows from (12), (13), and (11) that

$$w(f \oplus f_1 \oplus f_2) \leqslant \tfrac{1}{2} w(f \oplus f_1) \leqslant \tfrac{1}{4} w(f).$$

Using (13) and (11), we obtain

$$w(f_2) \leqslant \tfrac{3}{2} w(f \oplus f_1) \leqslant \tfrac{3}{4} w(f).$$

Repeating this procedure s times yields a sequence of functions f_i, $2 \leqslant i \leqslant s$, such that

$$f_i \geqslant f \oplus \left(\bigoplus_{j=1}^{i-1} f_j \right), \tag{15}$$

$$w(f_i) \leqslant \frac{3}{2} w\left(f \oplus \left(\bigoplus_{j=1}^{i-1} f_j \right) \right) \tag{16}$$

and for $w(f_i) > n^{c-1}$,

$$L(f_i) \leqslant \frac{\log \binom{4|D|}{w(f_i)}}{\log\log \binom{4|D|}{w(f_i)}}(1 + o(1)). \tag{17}$$

From (15) and (16), we have

$$w\left(f \oplus \left(\bigoplus_{j=1}^{i} f_j\right)\right) \leqslant \frac{1}{2} w\left(f \oplus \left(\bigoplus_{j=1}^{i-1} f_j\right)\right). \tag{18}$$

Using (18), (16), and induction on i, we easily derive the following inequalities:

$$w\left(f \oplus \left(\bigoplus_{j=1}^{i} f_j\right)\right) \leqslant \frac{1}{2} w\left(f \oplus \left(\bigoplus_{j=1}^{i-1} f_j\right)\right) \leqslant \frac{1}{2^i} w(f),$$

$$w(f_i) \leqslant \frac{3}{2} w\left(f \oplus \left(\bigoplus_{j=1}^{i-1} f_j\right)\right) \leqslant \frac{3}{2^i} w(f). \tag{19}$$

The parameter s is chosen so that f_s is the first function for which

$$w\left(f \oplus \left(\bigoplus_{i=1}^{s} f_i\right)\right) \leqslant w(f)/n.$$

Using (19), we get $s < \log n + 2$.

We now verify that

$$L\left(f \oplus \left(\bigoplus_{i=1}^{s} f_i\right)\right) \leqslant \frac{w(f)}{\log w(f)}(1 + o(1)). \tag{20}$$

Indeed, Lupanov's result mentioned above implies that

$$L\left(f \oplus \left(\bigoplus_{i=1}^{s} f_i\right)\right) \leqslant \frac{\log\binom{2^n}{\lfloor w(f)/n \rfloor}}{\log\log\binom{2^n}{\lfloor w(f)/n \rfloor}}(1 + o(1)). \tag{21}$$

Since $w(f) \geqslant n^c$ with $c \geqslant 3$, we have

$$\binom{2^n}{\lfloor w(f)/n \rfloor} \leqslant \left(\frac{3n2^n}{w(f)}\right)^{w(f)/n} \leqslant 2^{w(f)}. \tag{22}$$

Substituting (22) into (21) gives (20). On the other hand, whenever $w(f) < \frac{1}{10}|D| < 2^n$, we have

$$w(f) < \log\binom{4|D|}{w(f)}. \tag{23}$$

By definition

$$f' = f \oplus \left(\bigoplus_{i=1}^{s} f_i\right).$$

Then

$$f = f' \oplus \left(\bigoplus_{i=1}^{s} f_i\right)$$

and

$$L(f) \leqslant L(f') + \sum_{i=1}^{s} L(f_i) + s + 1. \tag{24}$$

Since $w(f')\leqslant w(f)/n$, we find, according to [2], that

$$L(f') < \log \binom{2^n}{\lfloor w(f)/n \rfloor} \Big/ \log\log \binom{2^n}{\lfloor w(f)/n \rfloor} (1+\mathrm{o}(1)) \leqslant \text{(see (20))}$$

$$< \frac{w(f)}{\log w(f)}(1+\mathrm{o}(1))$$

$$\leqslant \text{(see (23))} < \log \binom{4|D|}{w(f)} \Big/ \log\log \binom{4|D|}{w(f)} (1+\mathrm{o}(1)). \tag{25}$$

Using (19), the fact that $\varphi(x)=x/\log x$ increases whenever $x>e$, and the inequalities $i\leqslant s\leqslant \log n+2$ and

$$L(f_i)\leqslant (1+\mathrm{o}(1))\log \binom{4|D|}{w(f_i)} \Big/ \log\log \binom{4|D|}{w(f_i)},$$

we obtain

$$L(f_i) \leqslant (1+\mathrm{o}(1))\log \binom{4|D|}{\lfloor 3\cdot 2^{-i} w(f) \rfloor} \Big/ \log\log \binom{4|D|}{\lfloor 3\cdot 2^{-i} w(f) \rfloor}$$

$$< \frac{(1+\mathrm{o}(1))\log(4|D|2^i/w(f))^{3\cdot 2^{-i}w(f)}}{\log\log(4|D|/w(f))^{3\cdot 2^{-i}w(f)}}$$

$$= \frac{(1+\mathrm{o}(1))3\cdot 2^{-i}w(f)\log(4|D|2^i/w(f))}{\log(2^{-i}w(f))}. \tag{26}$$

It follows from (26) that

$$\sum_{i=1}^{s} L(f_i) < 3(1+\mathrm{o}(1))w(f)\sum_{i=1}^{s} \frac{2^{-i}\log(4|D|2^i/w(f))}{\log(2^{-i}w(f))}$$

$$= 3(1+\mathrm{o}(1))w(f)\left\{\sum_{i=1}^{s} \frac{2^{-i}\log(4|D|/w(f))}{\log(2^{-i}w(f))} + \sum_{i=1}^{s} \frac{i\,2^{-i}}{\log(2^{-i}w(f))}\right\}$$

$$= \frac{3(1+\mathrm{o}(1))w(f)\log(4|D|/w(f))}{\log w(f)}$$

$$\leqslant 3(1+\mathrm{o}(1))\log \binom{4|D|}{w(f)} \Big/ \log\log \binom{4|D|}{w(f)}. \tag{27}$$

Substituting (25) and (27) into (24), we arrive at the conclusion of Lemma 6. □

Theorem 4. *Let $c>3$, and let n be sufficiently large. Then there exist constants c_{10} and c_{11} such that*

$$c_{10}\frac{n^c}{\log n} \leqslant T_c(n) \leqslant c_{11}\frac{n^c}{\log n}.$$

Proof. *Lower bound.* Assume that the lower bound is not valid, i.e.,

$$T_c(n) = \mathrm{o}\left(\frac{n^c}{\log n}\right)$$

for some $c > 3$. This means that if a Boolean function f of n variables is equal to 1 on n^c tuples, then

$$T(f) < c_{12}\frac{n^c}{\log n}, \tag{28}$$

where c_{12} is a sufficiently small constant.

Let P be a program that computes f and satisfies (28). Let P' be an initial subprogram of P that computes f on all tuples whose numbers are not greater than 2^{n-1}. It follows from Lemma 3 and (28) that the number of distinct subprograms P' does not exceed $2^{c_{12}c_{13}n^c}$. Therefore, the number of such functions is at most

$$2^{c_{12}c_{13}n^c}\sum_{i=0}^{n^c}\binom{2^{n-1}}{i} < 2^{c_{12}c_{14}n^c}\binom{2^{n-1}}{n^c} < 2^{c_{12}c_{14}n^c-n^c}\binom{2^n}{n^c}.$$

Since the number of Boolean functions that take value 1 on n^c tuples is equal to $\binom{2^n}{n^c}$, we have $2^{c_{12}c_{14}n^c-n^c} \geqslant 1$. However, this inequality fails whenever $c_{12} < 1/2c_{14}$, a contradiction. Thus, $T_c(n) > c_{10}n^c/\log n$.

Upper bound. Let f be an arbitrary Boolean function depending on n variables and equal to 1 on n^c tuples, where $c \geqslant 3$. Let D be the domain consisting of the tuples on which f equals 1. In $\{0,1\}^n$, consider the set of all domains

$$M_1 = \{W_i \mid D \subset W_i,\ |W_i| = 25|D|\}$$

and partial Boolean functions $f_i : W_i \to \{0,1\}$ such that $f_i(x) = f(x)$ for $x \in W_i$, i.e., $f_i(x) = 1$ for $x \in D$ and $f_i(x) = 0$ for $x \in W \setminus D$.

By Lemma 6,

$$L(f_i) \leqslant \frac{4\log\binom{100|D|}{\lfloor 3w(f)/2\rfloor}}{\log\log\binom{100|D|}{\lfloor 3w(f)/2\rfloor}}(1+\mathrm{o}(1)) < \frac{45n^c}{c\log n}. \tag{29}$$

It follows from Lemma 2 and (29) that there exists a set consisting of at most

$$\binom{100|D|}{\lfloor 3w(f)/2\rfloor}^{4(1+\mathrm{o}(1))} = \binom{100|D|}{\lfloor 3|D|/2\rfloor}^{4(1+\mathrm{o}(1))}$$

circuits, each of size at most

$$4\log\binom{100|D|}{\lfloor 3|D|/2\rfloor}(1+\mathrm{o}(1));$$

moreover, for any function f_i, there exists a circuit in this set that implements this function. Since the number of distinct functions f_i is $\binom{2^n-|D|}{24|D|}$, there exists a circuit S_1

that simultaneously implements at least

$$\binom{2^n-|D|}{24|D|}\Big/\binom{100|D|}{\lfloor 3|D|/2\rfloor}^{4(1+o(1))}$$

distinct functions f_i, i.e., a function h_1 implemented by S_1 coincides with f on at least

$$\binom{2^n-|D|}{24|D|}\Big/\binom{100|D|}{\lfloor 3|D|/2\rfloor}^{4(1+o(1))}$$

domains in M_1. The set of tuples that belong to the union of these domains and are not contained in D is denoted by U_1. Put $q_i=|U_1|$. Clearly, $h_1(x)=0$ for any $x\in U_1$. Since U_1 contains at least

$$\binom{2^n-|D|}{24|D|}\Big/\binom{100|D|}{\lfloor 3|D|/2\rfloor}^{4(1+o(1))}$$

subsets of size $24|D|$, it follows that

$$\binom{2^n-|D|}{24|D|}\Big/\binom{100|D|}{\lfloor 3|D|/2\rfloor}^{4(1+o(1))}\leqslant\binom{q_1}{24|D|},$$

i.e.,

$$\frac{\binom{2^n-|D|}{24|D|}}{\binom{100|D|}{\lfloor 3|D|/2\rfloor}^{4(1+o(1))}\binom{q_1}{24|D|}}<1. \tag{30}$$

We next verify that (30) holds only if $q_1>2^{n-3}$. Indeed, for any $q_1\leqslant 2^{n-3}$ we have

$$\frac{\binom{2^n-|D|}{24|D|}}{\binom{100|D|}{\lfloor 3|D|/2\rfloor}^{4(1+o(1))}\binom{q_1}{24|D|}}$$

$$\geqslant\frac{\binom{2^n-|D|}{24|D|}}{\binom{100|D|}{\lfloor 3|D|/2\rfloor}^{4(1+o(1))}\binom{2^{n-3}}{24|D|}}$$

$$>\frac{4^{24|D|}}{\binom{100|D|}{\lfloor 3|D|/2\rfloor}^{4(1+o(1))}}>4^{24|D|}\left(\frac{\lfloor 3|D|/2\rfloor!}{(100|D|)^{\lfloor 3|D|/2\rfloor}}\right)^{4(1+o(1))}$$

$$>\left(\text{because } r!>\left(\frac{r}{e}\right)^r\right)>\frac{4^{24|D|}}{200^{6|D|(1+o(1))}}>1$$

which contradicts (30). Therefore, $q_1>2^{n-3}$.

Denote by R_1 the set of tuples in $\{0,1\}^n \setminus D$ on which the function h_1 is equal to 1. Consider the set of domains

$$M_2 = \{W_i \subset (R_1 \cup D) \mid D \subset W_i,\ |W_i| = 25|D|\}$$

and the partial Boolean functions $f_i : W_i \to \{0,1\}$ such that $f_i(x) = f(x)$ for $x \in W_i$. As in the previous case, we see that there exists a circuit S_2 implementing the Boolean function h_2 and such that

(a) S_2 simultaneously implements at least

$$\binom{R_1}{24|D|} \Big/ \binom{100|D|}{\lfloor 3|D|/2 \rfloor}^{4(1+o(1))}$$

distinct functions f_i;

(b) the number of tuples that belong to the union of these domains but are not contained in D is at least $\frac{1}{8}|R_1|$;

(c) $L(h_2) < 45n^c/c \log n$.

By R_2 denote the set of tuples $x \in R_1$ such that $h_2(x) = 1$. If $|R_2| > 2^n/n$, then we consider the set of domains

$$M_3 = \{W_i \subset (R_2 \cup D) \mid D \subset W_i,\ |W_i| = 25|D|\}$$

and the partial Boolean functions $f_i : W_i \to \{0,1\}$ such that $f_i(x) = f(x)$ for $x \in W_i$. As in the first case, we see that there exist a set R_3 and a function h_3 satisfying (a)–(c).

The domains $R_1, R_2, \ldots$ such that $R_1 \supset R_2 \supset \cdots$ and $|R_i| \leqslant \frac{7}{8}|R_{i-1}|$ are generated until we obtain a domain R_s such that $|R_s| < 2^n/n$. Eventually, we arrive at a sequence of functions $h_1, h_2, \ldots, h_{s-1}$ such that

$$L(h_i) < \frac{45n^c}{c \log n} \tag{31}$$

for any i, $1 \leqslant i \leqslant s-1$.

Now let us describe a program P computing the function f on an arbitrary tuple y. This program is constructed from circuits implementing the functions f and h_i and runs as follows. The function $h_1(y)$ is computed first. If $h_1(y) = 0$, then the program terminates and we set $f(y) = 0$. If $h_1(y) = 1$, then $h_2(y)$ is computed. If $h_2(y) = 0$, then the program terminates and we set $f(y) = 0$. If $h_2(y) = 1$, then $h_3(y)$ is computed, etc. If $h_{s-1}(y) = 1$, then $f(y)$ is computed by a program that models the circuit from [2] implementing a function f with a given number of ones.

We now find an upper bound on the average execution time of the program P. Using (31), we obtain

$$T(P) \leqslant \frac{1}{2^n}\left(\sum_{i=1}^{s-1} L(h_i) 2^n \left(\frac{7}{8}\right)^{i-1} + L(f) 2^n n^{-1}\right) \leqslant \mathrm{O}\left(\frac{n^c}{\log n}\right).$$

Theorem 4 is proved. □

5. Partial Boolean functions

Hereafter, we consider partial Boolean functions of n variables that are defined in restricted domains. The average execution time for such functions is defined in the same way as for completely defined Boolean functions, namely, by averaging over all 2^n tuples.

Let $T(n,d)=\max T(f)$, where the maximum is taken over all partial Boolean functions depending on n variables and defined in domains of size d.

Theorem 5. *If* $d > n\log n$, then

$$c_{15}\frac{d}{\log d}\leqslant T(n,d)\leqslant c_{16}\frac{d}{\log d}.$$

Proof. The upper bound follows from the corresponding results for the circuit complexity [1,3]. We now prove the lower bound. Consider the following two cases:
(I) $d\geqslant 2^{n-1}$;
(II) $d<2^{n-1}$.

Case 1. Denote by D the set of all tuples $(x_1,\ldots,x_n)$ in $\{0,1\}^n$ such that $x_1=1$. Let $\tilde{f}(x_2,\ldots,x_n)$ be a complete Boolean function such that

$$T(\tilde{f})=T(n-1).$$

Consider the function $f=\tilde{f}$ in D. For any program P computing f, we have

$$T(P)=2^{-n}\left(\sum_{x\in D}T_P(x)+\sum_{x\notin D}T_P(x)\right)\geqslant\frac{1}{2}2^{-(n-1)}\sum_{x\in D}T_P(x)$$
$$\geqslant\frac{1}{2}T(\tilde{f})=\frac{1}{2}T(n-1).$$

The assertion of the theorem now follows from Theorem 1.

Case 2. We prove the theorem by contradiction. Let f be an arbitrary partial Boolean function depending on n variables and defined in a domain of size d. Then

$$T(f)\leqslant\frac{c_{17}d}{\log d},\tag{32}$$

where c_{17} is a sufficiently small constant. Suppose that P is a program that computes the function f and satisfies (32). Let P' be an initial subprogram of P that computes f on all tuples x whose numbers $N_{P'}(x)$ do not exceed 2^{n-1}. Lemma 3 and (32) together imply that the number of such subprograms P' is at most $2^{c_{17}c_{18}d}$. Now we use P' to estimate the number R of Boolean functions that depend on n variables, have domains of size d, and satisfy (32). It is easy to see that

$$R\leqslant 2^{c_{12}c_{17}d}\left(\sum_{i=0}^{d}\binom{2^{n-1}}{d-i}\binom{2^{n-1}}{i}2^i\right).$$

Since

$$\frac{\binom{2^{n-1}}{i}2^i}{\binom{2^n}{i}} = \frac{2^{n-1}!(2^n-i)!2^i}{(2^{n-1}-i)!2^n!} = 2^i\prod_{j=0}^{i-1}\frac{2^{n-1}-j}{2^n-j} = \prod_{j=0}^{i-1}\frac{2^n-2j}{2^n-j} \leqslant 1,$$

it follows that

$$\binom{2^{n-1}}{i}2^i \leqslant \binom{2^n}{i}.$$

Therefore,

$$\sum_{i=0}^{d}\binom{2^{n-1}}{d-i}\binom{2^{n-1}}{i}2^i \leqslant \sum_{i=0}^{d}\binom{2^{n-1}}{d-i}\binom{2^n}{i} = \binom{3\cdot 2^{n-1}}{d}$$

and

$$R \leqslant 2^{c_{12}c_{17}d}\binom{3\cdot 2^{n-1}}{d}. \tag{33}$$

Furthermore, the number of such partial Boolean functions is equal to $2^d\binom{2^n}{d}$. Now, put

$$\varphi(n,d) = \binom{3\cdot 2^{n-1}}{d} \Big/ \left(2^d\binom{2^n}{d}\right);$$

we see that

$$\varphi(n,d) < \left(\frac{25}{27}\right)^d \tag{34}$$

for any d, $1 \leqslant d \leqslant 2^{n-1}$. Indeed, the Stirling formula yields

$$\begin{aligned}
\varphi(n,d) &= \frac{(3\cdot 2^{n-1})!(2^n-d)!}{(3\cdot 2^{n-1}-d)!2^n!2^d} < \frac{3^{3\cdot 2^{n-1}}2^{3(n-1)2^{n-1}}(2^n-d)^{2^n-d}}{(3\cdot 2^{n-1}-d)^{3\cdot 2^{n-1}-d}2^{n2^n}2^d} \\
&= \frac{3^{3\cdot 2^{n-1}}2^{3(n-1)2^{n-1}}2^{n(2^n-d)}}{3^{3\cdot 2^{n-1}-d}2^{(n-1)(3\cdot 2^{n-1}-d)}2^{n2^n}2^d}\cdot\frac{(1-d/2^n)^{2^n-d}}{(1-d/3\cdot 2^{n-1})^{3\cdot 2^{n-1}-d}} \\
&= \left(\frac{3}{4}\right)^d\left(1-\frac{d}{2^n}\right)^{2^n-d}\Big/\left(1-\frac{d}{3\cdot 2^{n-1}}\right)^{3\cdot 2^{n-1}-d} \\
&= \left(\left(3-\frac{d}{2^{n-1}}\right)\Big/\left(4-\frac{4d}{2^n}\right)\right)^d\left(\left(1-\frac{d}{2^n}\right)^2\Big/\left(1-\frac{d}{3\cdot 2^{n-1}}\right)^3\right)^{2^{n-1}}.
\end{aligned}$$

Clearly, $(3-d/2^{n-1})/(4-4d/2^n)$ increases on d and does not exceed 1 in the interval $[1,2^{n-1}]$, whereas $(1-d/2^n)^2/(1-d/(3\cdot 2^{n-1}))^3$ decreases. Therefore,
(a) if $d\in[1,3\cdot 2^{n-2}]$ then

$$\varphi(n,d)<\left(\left(3-\frac{d}{2^{n-1}}\right)\Big/\left(4-\frac{4d}{2^n}\right)\right)^d$$
$$<\left(\left(3-\frac{3\cdot 2^{n-3}}{2^{n-1}}\right)\Big/\left(4-\frac{4\cdot 3\cdot 2^{n-3}}{2^n}\right)\right)^d<\left(\frac{9}{10}\right)^d;$$

(b) if $d\in[3\cdot 2^{n-3},2^{n-1}]$ then

$$\varphi(n,d)<\left(\left(1-\frac{d}{2^n}\right)\Big/\left(1-\frac{d}{3\cdot 2^{n-1}}\right)^3\right)^d$$
$$<\left(\left(1-\frac{3\cdot 2^{n-3}}{2^n}\right)\Big/\left(1-\frac{3\cdot 2^{n-3}}{3\cdot 2^{n-1}}\right)^3\right)^d\leqslant\left(\frac{25}{27}\right)^d.$$

Thus, (33) holds whenever $1\leqslant d\leqslant 2^{n-1}$.

It follows from (33) and (34), that

$$\frac{R}{2^d\binom{2^n}{d}}<2^{c_{12}c_{17}d}\left(\frac{25}{27}\right)^d.$$

Therefore, for c_{12} sufficiently small,

$$R\Big/\left(2^d\binom{2^n}{d}\right)<1,$$

i.e., R is smaller than the number of all partial Boolean functions depending on n variables and defined in domains of size d. This means that the assumption above fails. Theorem 5 is proved. □

Acknowledgements

The author is grateful to O.B. Lupanov and Yu.V. Tarannikov, who read the first version of the paper and made a number of helpful remarks.

References

[1] A.E. Andreev, On circuit complexity of partial Boolean functions, Diskret. Mat. 1(4) (1989) 36–45 (in Russian).
[2] O.B. Lupanov, An approach to the synthesis of control systems; the principle of local coding, Problemy Kibernetiki, Moscow, 14 (1965) 31–110 (in Russian).
[3] L.A. Sholomov, On the implementation of partial Boolean functions by circuits, Problemy Kibernetiki, Moscow, 21 (1969) 215–226 (in Russian).